# Not All FPRASs are Equal: Demystifying FPRASs for DNF-Counting

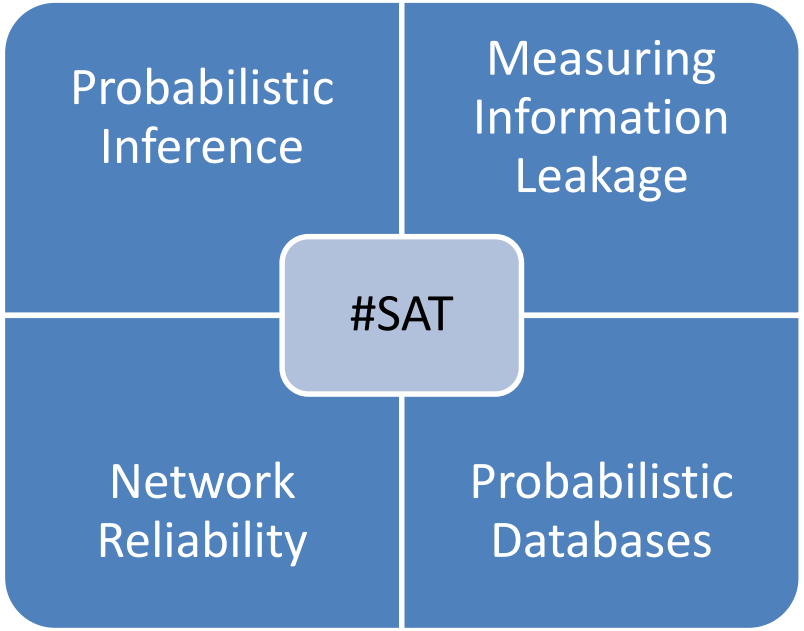
Kuldeep S. Meel[1], **Aditya A. Shrotri[2]**, Moshe Y. Vardi[2]

[1] School of Computing,
National University of Singapore

[2] Department of Computer Science,
Rice University

# Beyond SAT: #SAT

- SAT: *Does there exist a satisfying assignment?*
- #SAT: *How many satisfying assignments*?
  - Complexity: #P-Complete (contains entire polynomial hierarchy)
  - In Practice: Harder than SAT



# The Disjunctive Normal Form

- $F = (\neg X_1 \wedge X_2) \vee (X_2 \wedge X_3 \wedge X_4) \vee (X_1 \wedge X_3 \wedge \neg X_5)$
  - Disjunction of Cubes
- DNF-SAT is in P
- #DNF is #P-Complete [Valiant, ’79]
- Need to Approximate!

# Approximate DNF-Counting

**Input:** DNF Formula F

Tolerance $\varepsilon$ $0 < \varepsilon < 1$

Confidence $\delta$ $0 < \delta < 1$

**Output:** Approximate Count C s.t.

$$\Pr[\#F \cdot (1-\varepsilon) < C < \#F \cdot (1+\varepsilon)] > 1-\delta$$

# Fully Polynomial Randomized Approximation Scheme

**Input:** DNF Formula F

Tolerance $\varepsilon$ $0 < \varepsilon < 1$

Confidence $\delta$ $0 < \delta < 1$

**Output:** Approximate Count C s.t.

$$\Pr[\#F \cdot (1-\varepsilon) < C < \#F \cdot (1+\varepsilon)] > 1-\delta$$

**Challenge:** Design a poly(m, n, $\frac{1}{\epsilon}$, $\log(\frac{1}{\delta})$) time algorithm

where m = #cubes n = #vars

# Paradigms for #DNF FPRAS

- Monte Carlo Sampling
  - Best complexity: $O(m \cdot n \cdot log(\frac{1}{\delta}) \cdot \frac{1}{\varepsilon^2})$

- Hashing – based
  - Best complexity: $\tilde{O}(m \cdot n \cdot log(\frac{1}{\delta}) \cdot \frac{1}{\varepsilon^2})$

# Motivation

1. Can hashing – based approach have same complexity as Monte Carlo?
2. How do the various algorithms compare empirically?

# Our Contribution

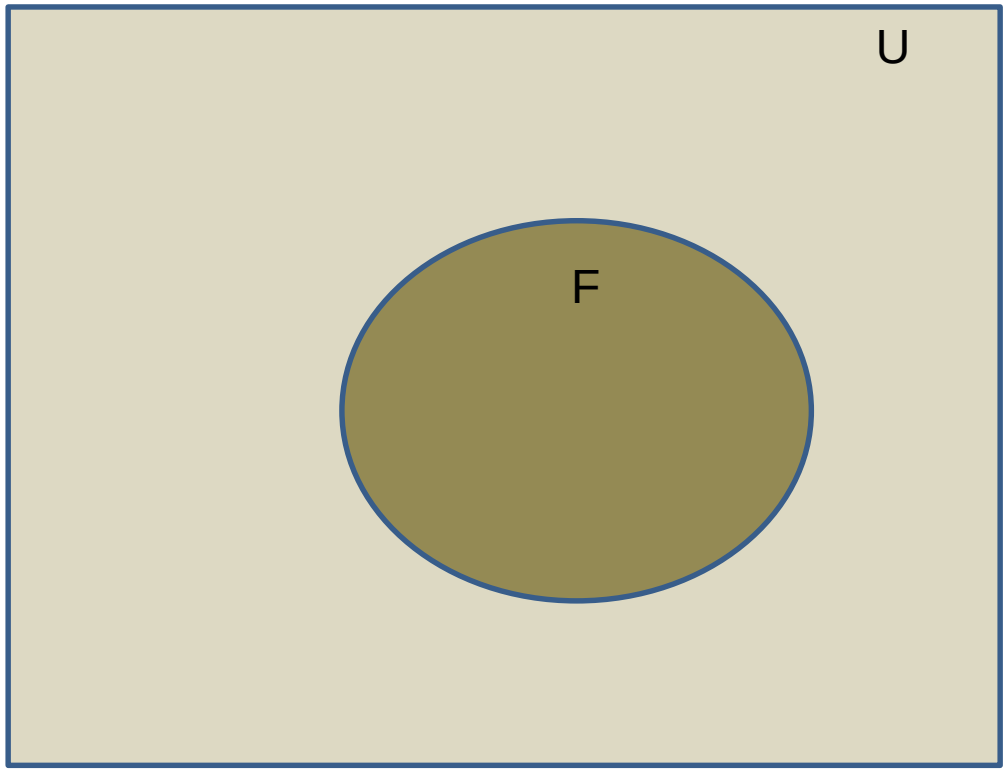
1. Can hashing – based approach have same complexity as Monte Carlo?
2. How do the various algorithms compare empirically?

**In this work:**

- Improved complexity of hashing – based algorithm
  - Improves practical performance

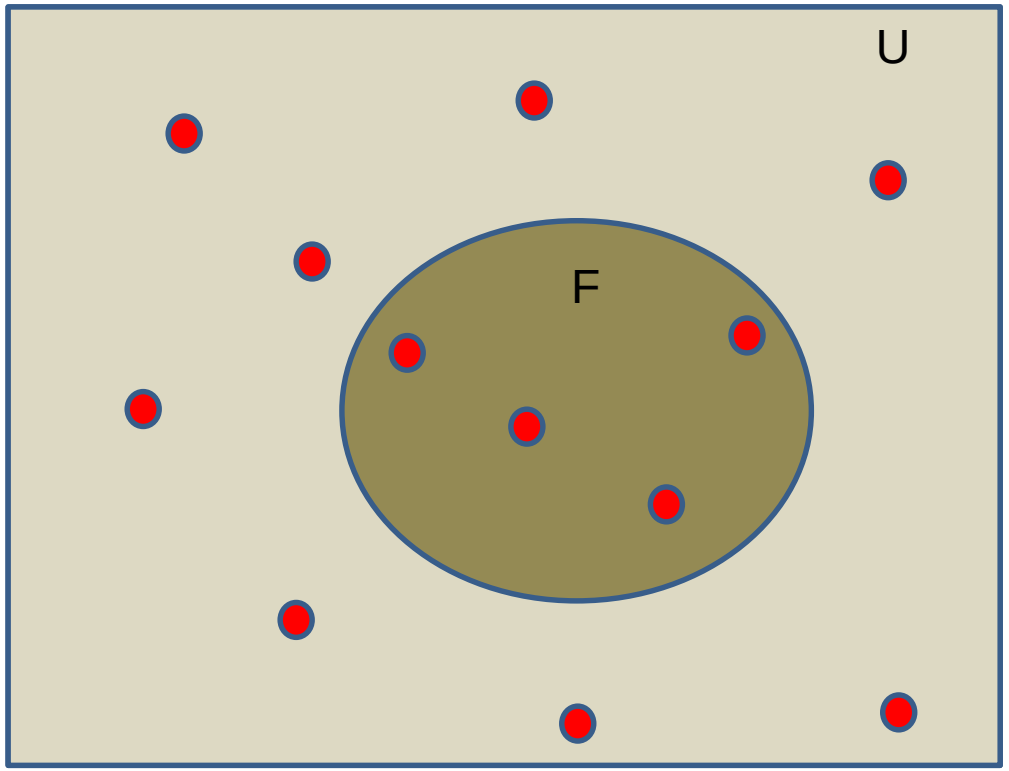
- First comprehensive empirical evaluation of FPRASs for #DNF
  - Much more nuanced than complexity analysis alone

# Method 1: Monte Carlo Sampling



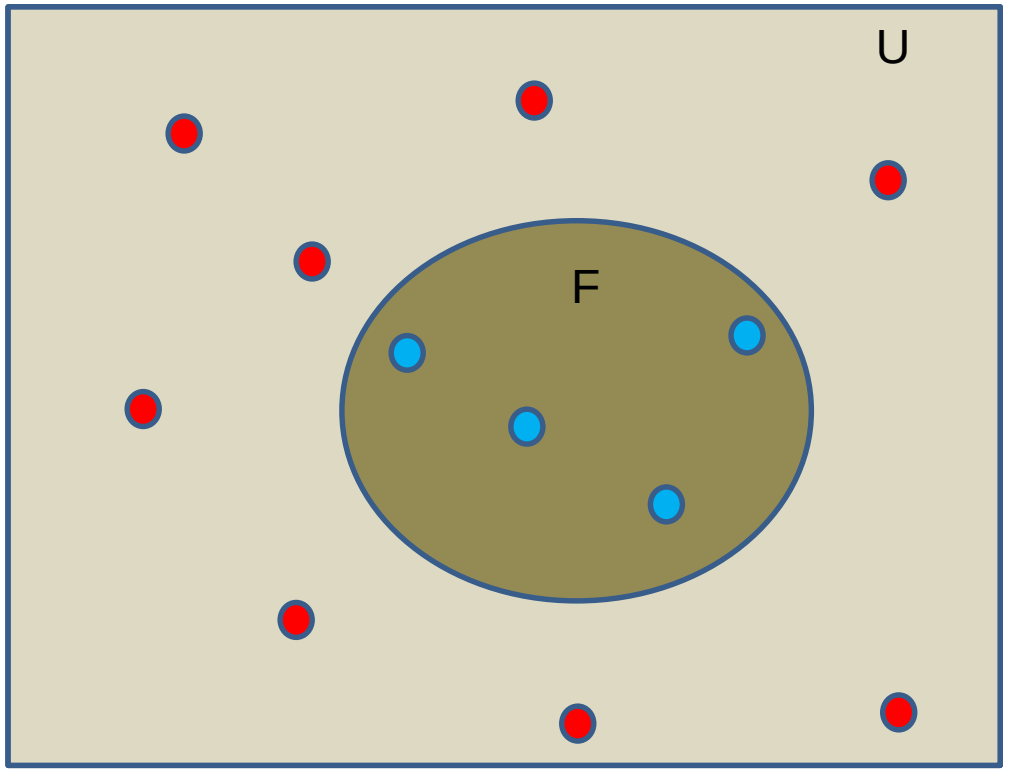

- Draw independent samples from U

# Method 1: Monte Carlo Sampling



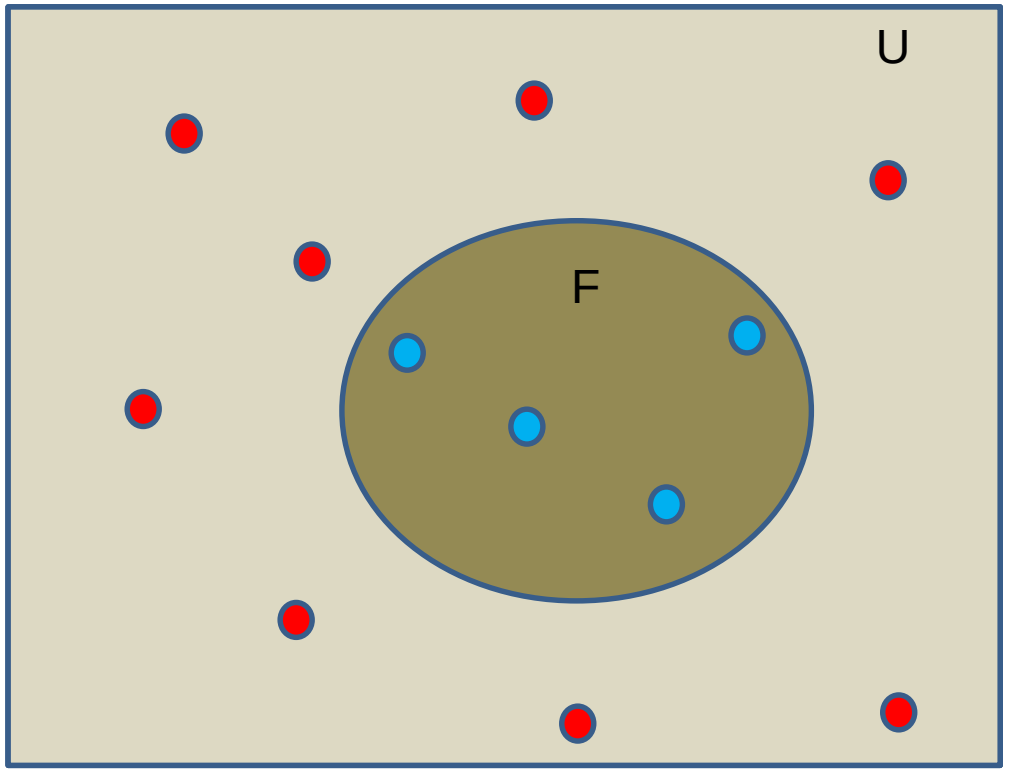

- Draw independent samples from U

# Method 1: Monte Carlo Sampling



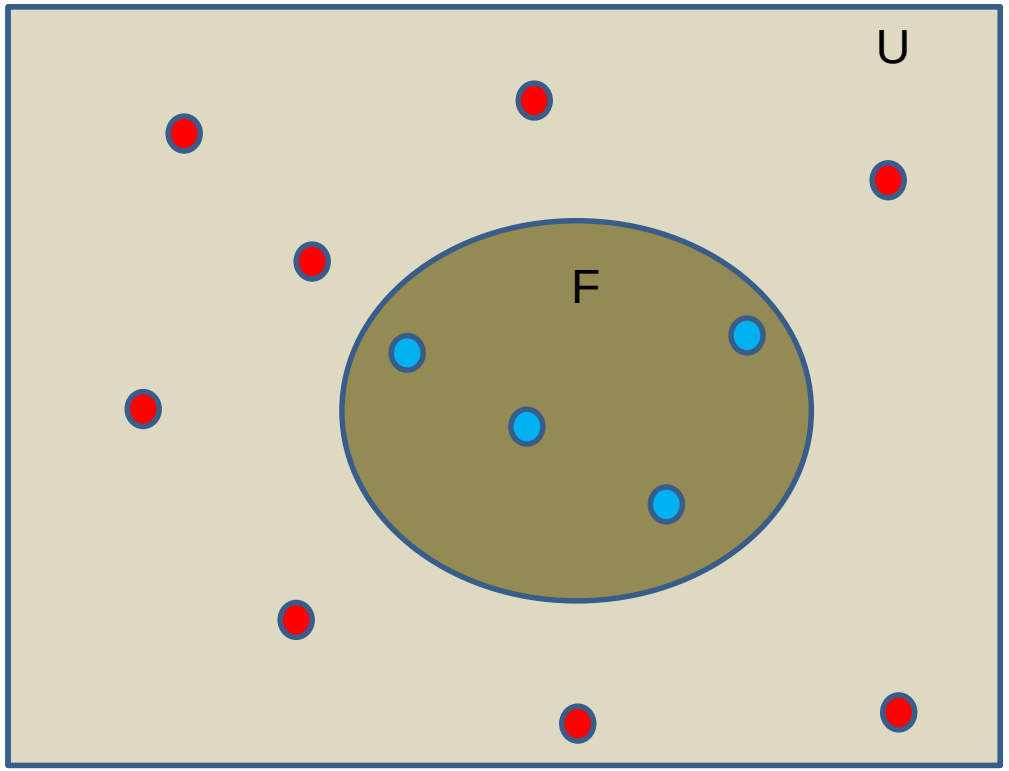

- Draw independent samples from U
- Count samples that satisfy F

# Method 1: Monte Carlo Sampling




- Draw independent samples from U
- Count samples that satisfy F
- $\#F \approx \frac{\#\bullet}{\#\bullet + \#\bullet} * |U|$

# Method 1: Monte Carlo Sampling

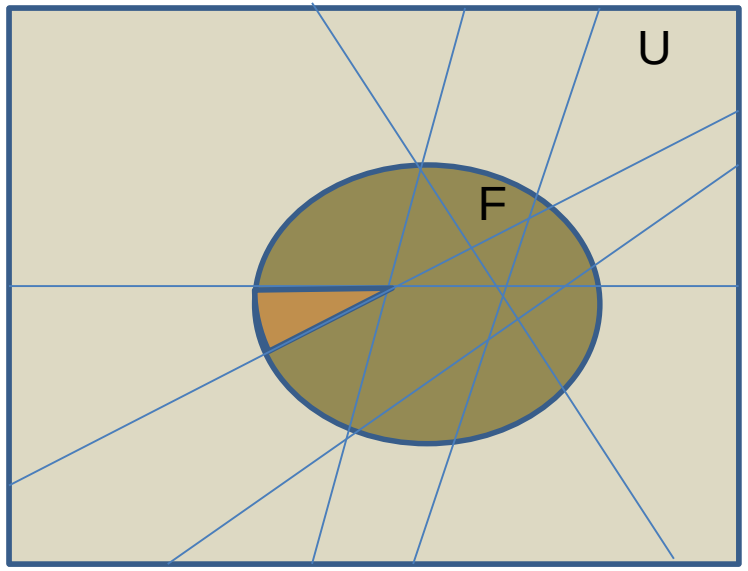



- Draw independent samples from U
- Count samples that satisfy F
- $\#F \approx \frac{\#\bullet}{\#\bullet + \#\bullet} * |U|$
- **Requirement:** $\frac{\#F}{|U|}$ is large
  - Solution density plays crucial role

# Monte Carlo FPRAS

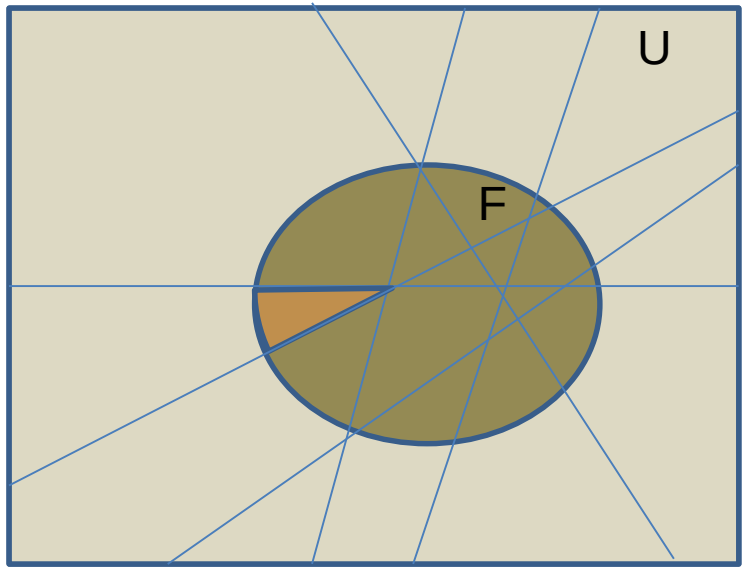
- KL Counter [Karp, Luby, ‘83]
  - Insight: Transform solution space
- KLM Counter [Karp et al., ‘89] $O(m \cdot n \cdot log(\frac{1}{\delta}) \cdot \frac{1}{\varepsilon^2})$
  - Insight: Sample using geometric distribution
- Vazirani Counter [Vazirani, ’13]
  - Insight: Low variance $\Longrightarrow$ fewer samples

# Method 2: Hashing-Based




- Partition U into small cells
- Count solutions in a random cell '$C_{cell}$'
- #F ≈ $C_{cell}$ * no. of cells

# Method 2: Hashing-Based




- Partition U into small cells
- Count solutions in a random cell '$C_{cell}$'
- #F ≈ $C_{cell}$ * no. of cells
- **Requirements:**
  - Roughly equal solutions in each cell
  - Right size of cell

# Method 2: Hashing-Based

- **Requirement:** Roughly Equal Solutions in each cell

- **Solution:** Hash Functions
  - Conjunction of random XOR formulas
  - h = $H_1 \wedge H_2 \wedge \ \ldots \wedge H_i$
  - h: $2^n \rightarrow 2^i$

# Method 2: Hashing-Based

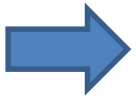
- **Requirement:** Right size cell

- **Solution:** Count up to threshold $T$
  - $T$ depends on $\varepsilon$ and $\delta$
  - $C_{cell} > T \implies$ cell too large
    - Increase no. of constraints $i$
  - Find $i$ such that $C_{cell} \leq T$

# Forward Search

- **Requirement:** Right size partition

- **Solution:** Count up to threshold $T$
  - Find $i$ such that $C_{cell} \leq T$



| $i$ | 0 |
|---|---|
| $C_{cell}$ | $C_{cell} > T$ |
| Cell size | Too large |

# Forward Search

- **Requirement:** Right size partition

- **Solution:** Count up to threshold $T$
  - Find $i$ such that $C_{cell} \leq T$

| $i$ | 0 | 1 |
|---|---|---|
| $C_{cell}$ | $C_{cell} > T$ | $C_{cell} > T$ |
| **Cell size** | Too large | Too large |

# Forward Search

- **Requirement:** Right size partition

- **Solution:** Count up to threshold $T$
  - Find $i$ such that $C_{cell} \leq T$

| $i$ | 0 | 1 | 2 |
|---|---|---|---|
| $C_{cell}$ | $C_{cell} > T$ | $C_{cell} > T$ | $C_{cell} > T$ |
| Cell size | Too large | Too large | Too large |

# Forward Search

- **Requirement:** Right size partition

- **Solution:** Count up to threshold $T$
  - Find $i$ such that $C_{cell} \leq T$

| $i$ | 0 | 1 | 2 | … | $l-1$ | $l$ |
|---|---|---|---|---|---|---|
| $C_{cell}$ | $C_{cell} > T$ | $C_{cell} > T$ | $C_{cell} > T$ | … | $C_{cell} > T$ | $C_{cell} \leq T$ |
| Cell size | Too large | Too large | Too large | … | Too large | Perfect |

  - #F ≈ $C_{cell} * 2^{l}$

# Hashing-Based FPRAS

1. DNFApproxMC [Chakraborty et al., '16]
   - Insight: Satisfiability of partitioned DNF formulas is polytime

2. SymbolicDNFApproxMC [Meel et al., '17] $\tilde{O}(m \cdot n \cdot log(\frac{1}{\delta}) \cdot \frac{1}{\varepsilon^2})$
   - Insight: Symbolic hash constraints applied to transformed space

# Forward Search

- **Requirement:** Right size partition

- **Solution:** Count up to threshold $T$
  - Find $i$ such that $C_{cell} \leq T$

| $i$ | 0 | 1 | 2 | … | $l-1$ | $l$ |
|---|---|---|---|---|---|---|
| $C_{cell}$ | $C_{cell} > T$ | $C_{cell} > T$ | $C_{cell} > T$ | … | $C_{cell} > T$ | $C_{cell} \leq T$ |
| **Cell size** | Too large | Too large | Too large | … | Too large | Perfect |

  - #F ≈ $C_{cell} * 2^{l}$

Drawback: Re-enumerates solutions multiple times

# Reverse Search

- Insight: Search in reverse direction

| $i$ | 0 | 1 | 2 | … | $l-1$ | $l$ | … | n-1 | n |
|---|---|---|---|---|---|---|---|---|---|
| $C_{cell}$ | $C_{cell} > T$ | $C_{cell} > T$ | $C_{cell} > T$ | … | $C_{cell} > T$ | $C_{cell} \leq T$ | … | $C_{cell} \leq T$ | $C_{cell} \leq T$ |
| **Cell size** | Too large | Too large | Too large | … | Too large | Perfect | … | Too small | Too small |

- Advantages:
  - Each solution is enumerated exactly once
  - $l$ is close to $n$ for DNF formulas

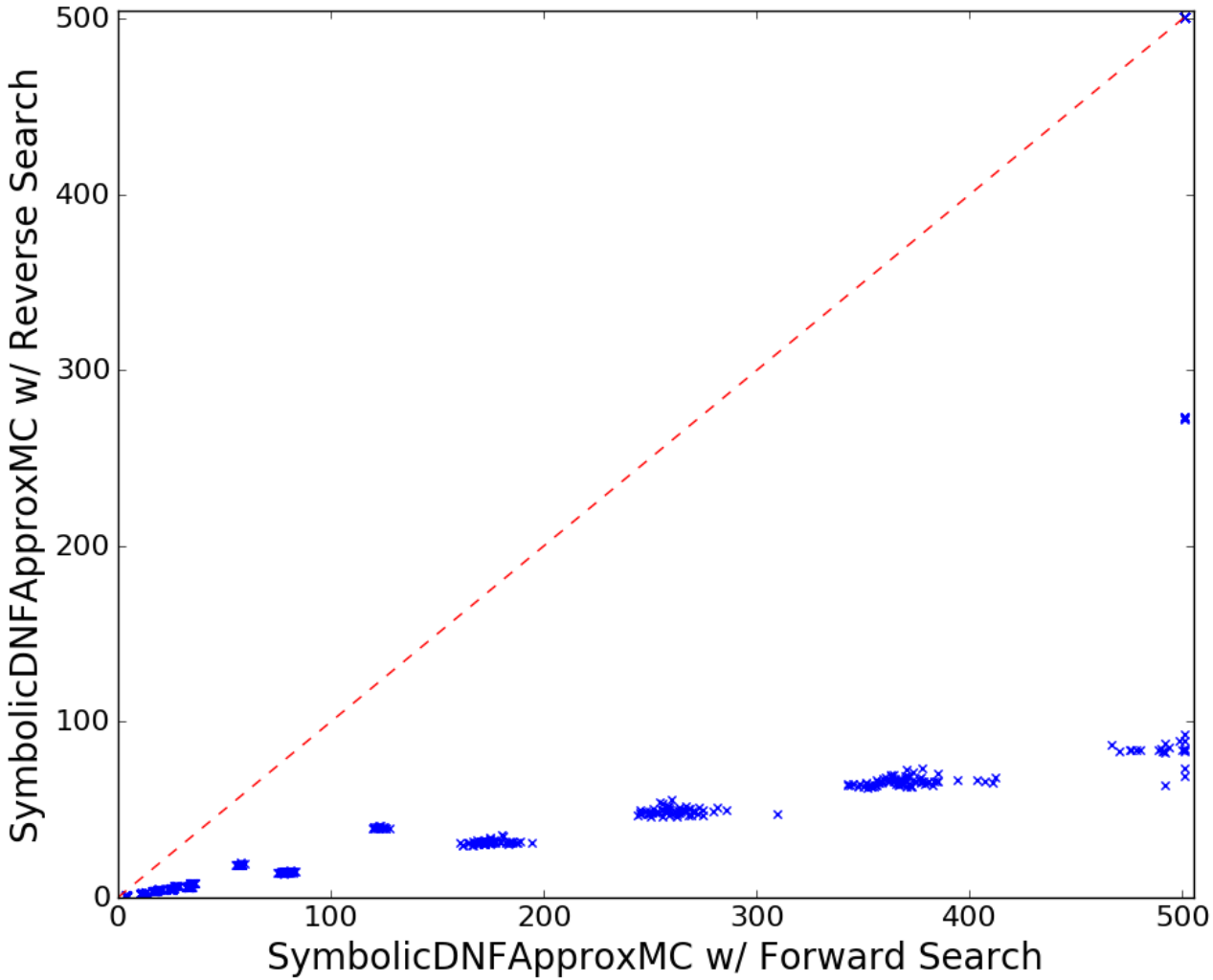
# Reverse Search

- Insight: Search in reverse direction

| $i$ | 0 | 1 | 2 | … | $l-1$ | $l$ | … | n-1 | n |
|---|---|---|---|---|---|---|---|---|---|
| $C_{cell}$ | $C_{cell} > T$ | $C_{cell} > T$ | $C_{cell} > T$ | … | $C_{cell} > T$ | $C_{cell} \leq T$ | … | $C_{cell} \leq T$ | $C_{cell} \leq T$ |
| Cell size | Too large | Too large | Too large | … | Too large | Perfect | … | Too small | Too small |

- Advantages:
  - Each solution is enumerated exactly once
  - $l$ is close to $n$ for DNF formulas
- **Theorem:** The complexity of SymbolicDNFApproxMC with Reverse Search is $O\left(m \cdot n \cdot log\left(\frac{1}{\delta}\right) \cdot \frac{1}{\varepsilon^2}\right)$

# Reverse Search vs. Forward Search




900+ benchmarks

- **X-axis:** Time taken by Binary Search
- **Y-axis:** Time taken by Reverse Search

Reverse Search is 4-5 times faster!

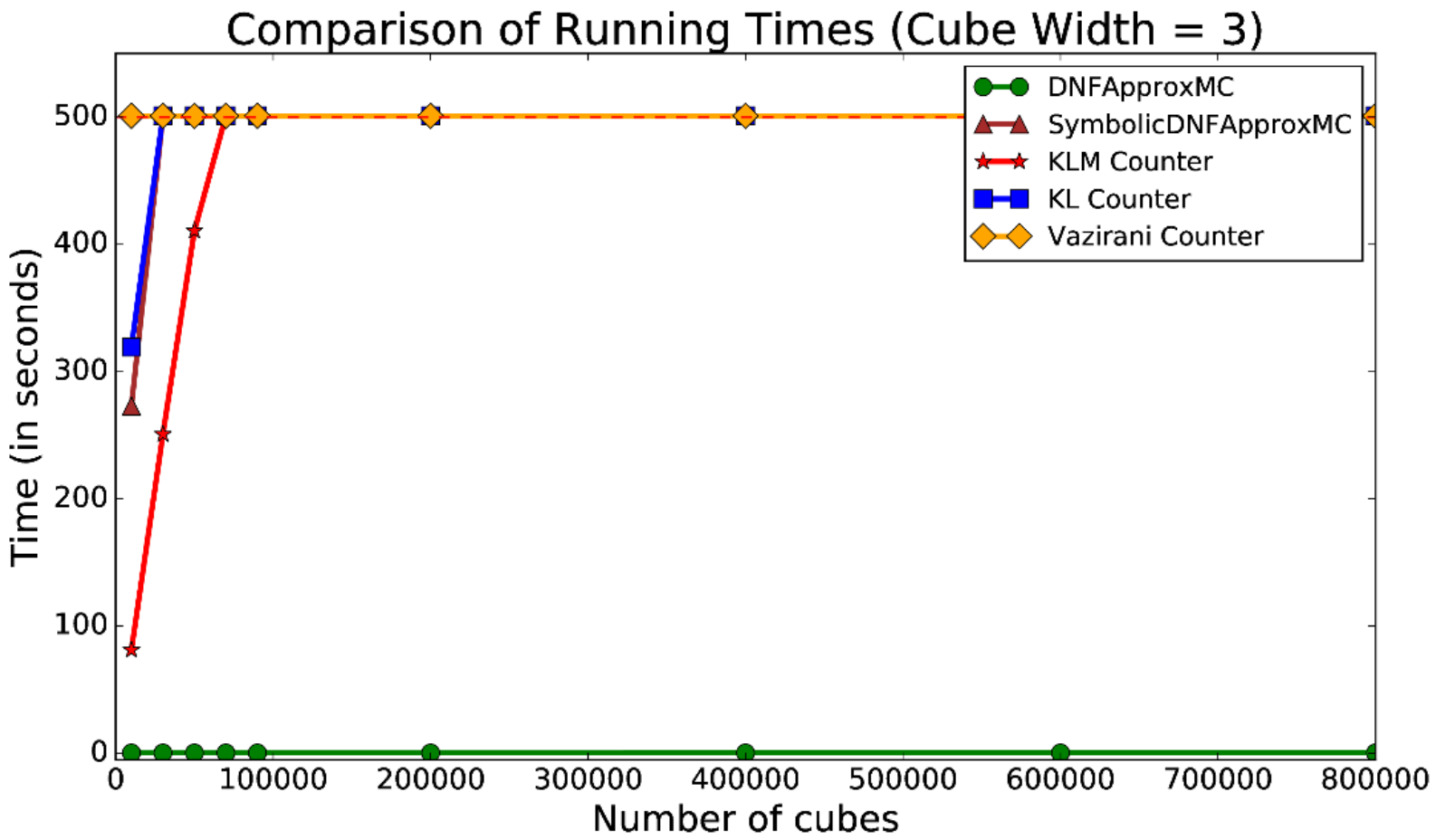
# Experiments: Algorithms

- Monte Carlo – Based
  - KL Counter $O(m^2 \cdot n \cdot log(\frac{1}{\delta}) \cdot \frac{1}{\varepsilon^2})$
  - Vazirani Counter $O(m^2 \cdot n \cdot log(\frac{1}{\delta}) \cdot \frac{1}{\varepsilon^2})$
  - KLM Counter $O(m \cdot n \cdot log(\frac{1}{\delta}) \cdot \frac{1}{\varepsilon^2})$
- Hashing – Based
  - DNFApproxMC $O(m \cdot n^2 \cdot log(\frac{1}{\delta}) \cdot \frac{1}{\varepsilon^2})$
  - SymbolicDNFApproxMC $O(m \cdot n \cdot log(\frac{1}{\delta}) \cdot \frac{1}{\varepsilon^2})$

# Runtime Variation

1. **Runtime Variation**: How does the running time of the algorithms vary across different benchmarks?
   - Parameters
     - # vars = 100,000
     - # cubes $\in [\ 10^4, 8 \times 10^6\ ]$
     - Cube width $\in \{\mathbf{3}, 13, \mathbf{23}, 33, 43\}$
   - 20 random benchmarks for each setting of parameters

# Runtime Variation




- **X-axis:** Number of cubes
- **Y-axis:** Median run time over 20 instances

- Timeout: 500 sec

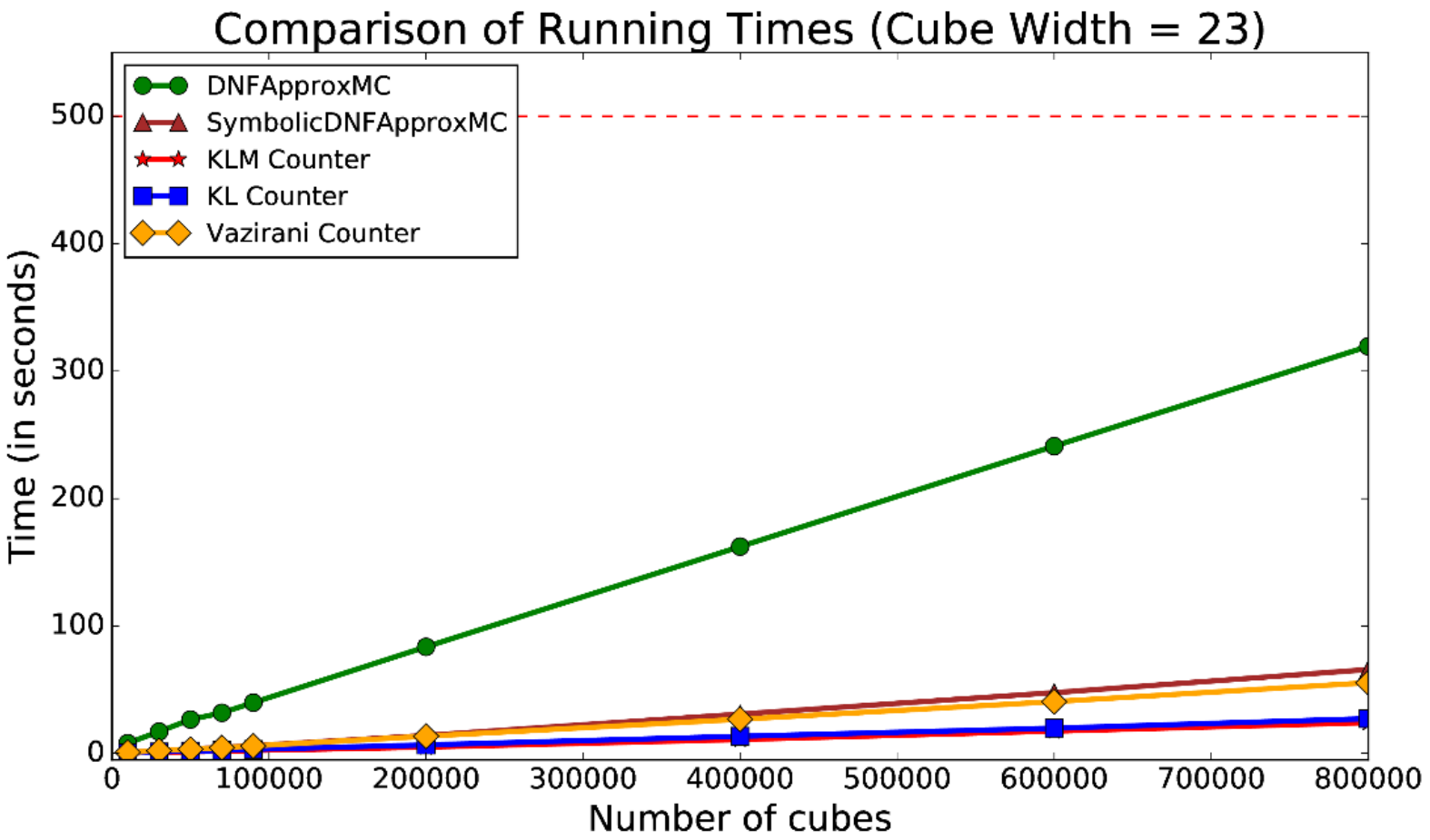
**Observations:**

- DNFApproxMC performs very well
- All other algorithms quickly timeout

**Reason:** Low density of solutions

- Monte Carlo algorithms and SymbolicDNFApproxMC are sensitive to solution density

# Runtime Variation




- **X-axis:** Number of cubes
- **Y-axis:** Median run time over 20 instances

- Timeout: 500 sec

**Observations:**

- DNFApproxMC performance degrades gracefully
- Performance of other algorithms improves dramatically

**Reason:** Density of solutions increases with cube-width

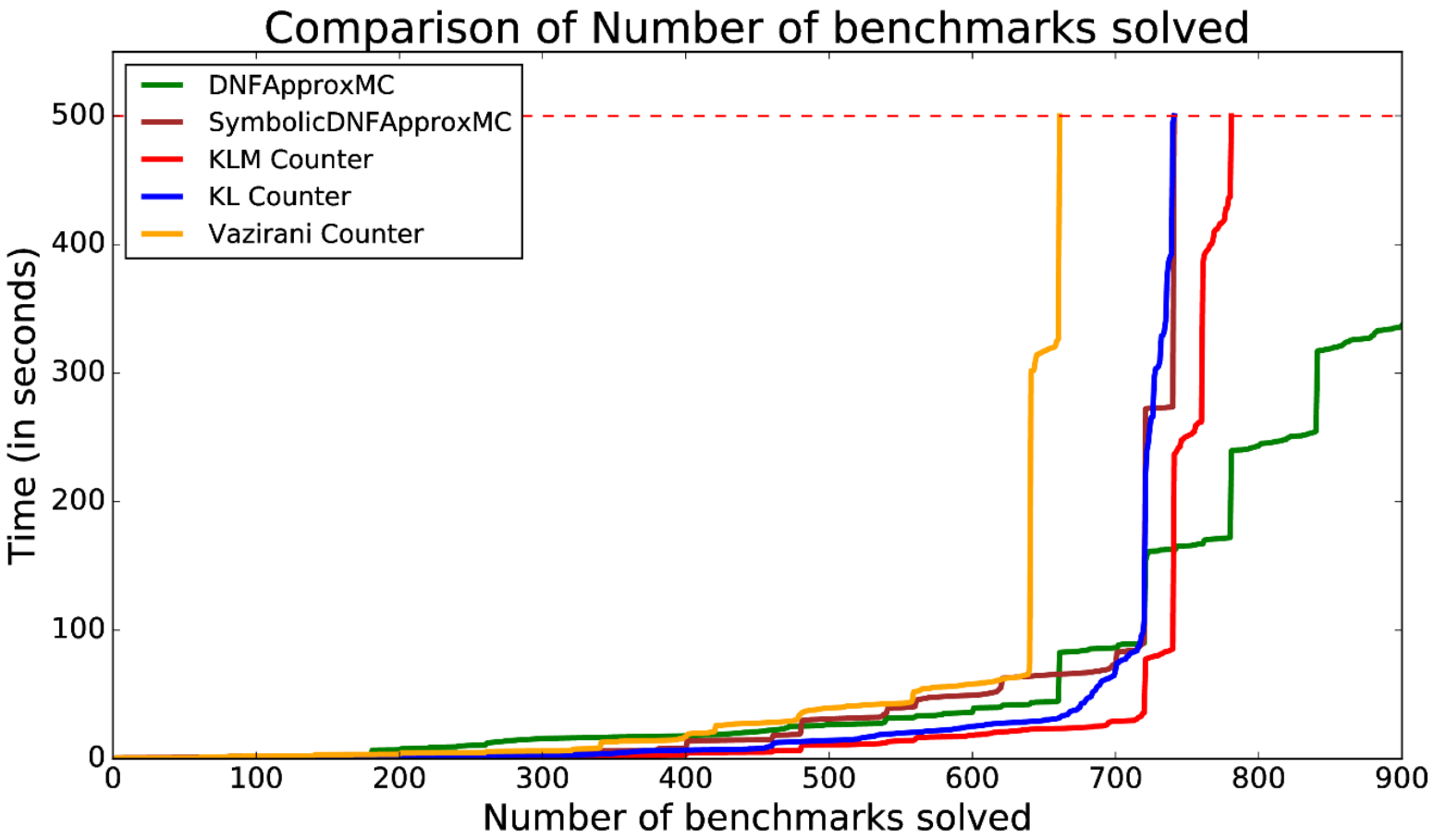
# Total Benchmarks Solved

2. **Total Benchmarks Solved**: How many benchmarks can the algorithms solve overall?

- Measures raw problem solving ability

- Same 900 benchmarks as **Runtime Variation**

# Total Benchmarks Solved




- Point (X,Y) represents X number of benchmarks out of 900 were solved in Y seconds or less

**Observations:**

- DNFApproxMC does not timeout on any formulas

**Reason:** Efficient data structures mitigate dependence on solution density

# Other Experiments

1. **Runtime Variation**: How does the running time of the algorithms vary across different benchmarks?
2. **Total Benchmarks Solved**: How many benchmarks can the algorithms solve overall?
3. **Accuracy**: How accurate are the counts returned by the algorithms?
4. $\epsilon - \delta$ **Scalability:** How do the algorithms scale with the input tolerance and confidence?

# Summary

- **Theory:**
  - Introduced Reverse Search for hashing – based counting
  - Improved the complexity of hashing FPRAS by polylog factors
    - 4 – 5 times speedup in practice

# Summary

- **Empirical Evaluation**
  - Presented nuanced picture
    - Different algorithms are well suited for different formula types
  - Algorithm with poor time complexity (DNFApproxMC) is most robust and solves largest number of benchmarks
  - Takeaways for practitioners:
    - High solution density $\Longrightarrow$ use KLM Counter
    - Low or unknown solution density $\Longrightarrow$ use DNFApproxMC

# Future Directions

- New counter that is robust as DNFApproxMC and fast as KLM Counter?
  - Portfolio of algorithms approach
- Real-world adoption of techniques
  - Break vicious cycle
- Study effect of Reverse Search on CNF and SMT Counting

# Backup Slides

# Experiments: Accuracy

| Algorithm | Mean Error | Max Error |
|---|---|---|
| DNFApproxMC | 0.09 | 0.36 |
| SymbolicDNFApproxMC | 0.21 | 0.42 |
| KLM Counter | 0.11 | 0.55 |
| KL Counter | 0.007 | 0.20 |
| Vazirani Counter | 0.001 | 0.04 |

# Experiments: $\epsilon - \delta$ Scalability